Welcome to Tetris! In this multi-part assignment, you will be building the Tetris game (with lots of help provided). It should take you approximately one week to complete the entire project.

This is another "Pair Programming" Assignment

While working on this project, you will use the "Pair Programming" approach. In pair programming, two programmers share one computer. One student is the "driver," who controls the keyboard and mouse. The other is the "navigator," who observes, asks questions, suggests solutions, and thinks about slightly longer-term strategies. You should switch roles about every 20 minutes. For this assignment, each partner should submit their own work.

Be sure that both of you have duplicate saved copies of your work each day. You need something to work with tomorrow if your partner is absent.

---

acknowledgments to Dave Feinberg for this assignment
(and his hilarious sense of humor!)

Start by familiarizing yourself with the following three classes and their associated methods.

Background: `Location`

`Location` objects represent a specific location on a 2-dimensional grid, and they keep track of the row and column of their location. Two `Location` objects are equivalent if their row and column values are the same.

`Location` class

```
public Location(int row, int col)
public int getRow()
public int getCol()
public boolean equals(Location other)
public String toString()
```

Background: `Grid`

The `Grid` class represents a 2-dimensional grid where the game will be played. It will allow us to place blocks into the grid at specific locations. Below is a summary of the constructor and other methods used in the `Grid` class.

`Grid` class

```
public Grid(int rows, int cols)
public int getNumRows()
public int getNumCols()
public Block get(Location location)
public Block put(Location location, Block block)
public boolean isValid(Location location)
public Block remove(Location location)
```

Background: `Block`

The first kind of thing we will place in our grid is a colored `Block`, an object which keeps track of its color, location, and the `Grid` that contains it. When the `Block` is not in a grid, its `grid` and `location` instance variables are both `null`. When the `Block` is in a grid, its `location` instance variable must match the `Block`'s location within the `Grid` associated with its `grid` instance variable. *This means that both the `Block` and its `grid` are keeping track of the `Block`'s location.* It is up to the `Block` class to keep these consistent.

The following example shows how to place a `Block` in a `Grid`.

```
Grid grid;
grid = new Grid(3, 2);
Block block = new Block();
block.putSelfInGrid(grid, new Location(2, 0));
```

Here is a summary of the `Block` class.

`Block` class

```
public Block()
public Color getColor()
public void setColor(Color newColor)
public Grid getGrid()
public Location getLocation()
public void putSelfInGrid(Grid grid, Location location)
public void removeSelfFromGrid()
public void moveTo(Location newLocation)
public String toString()
```

Time to get started with coding. Download, extract, and save the Tetris project files to your computer. Note that this project has nine files, with three of them listed as 'no-source'. Most files will not compile at this point, but you will soon remedy this situation.

Exercise 0.0 Location, Location, Location

Begin by writing a `Location` class from scratch according to the specifications listed above.

Your task is to satisfy the dreaded Grid Monster, who will test every aspect of your code. To complete this section, your code must pass each of the Grid Monster's tests. But beware! The Grid Monster has a nasty habit of insulting any code that displeases him (or her). Not every class in this project will compile at this point but `Location` must be compiled for the Grid Monster to test it.

When you have finished writing (and compiling) the `Location` class, run `GridMonster` to check your work before you continue with the next exercise. Once the `Location` class is correct, you should be able to compile all of the classes in this project.

Exercise 0.1: Coder's Block

Ah, but the Grid Monster is still not done with you yet. While the `Grid` class is already completed for you, you must now finish the incomplete methods in the `Block` class to continue. Most methods are easy to complete, but if you get stuck on some of the more difficult ones, follow the directions in the comments.

When you have completed the `Block` class, it is time to see if you have completed everything it takes to beat the dreaded Grid Monster! Go ahead and rerun the `GridMonster` class. Your code must pass each of the Grid Monster's tests to complete this part of the assignment. Good luck!

When you are finished, please submit in Schoology a screenshot of the `GridMonster` success message for Tetris – Part 0.